Advanced Operating Systems
Lecture notes

Dongho Kim
Tatyana Ryutov
University of Southern California
Information Sciences Institute

Administration

- Instructors
  - Dr. Dongho Kim
  - Dr. Tatyana Ryutov
- TA
  - Chansook Lim
- Office hours
  - Drs. Kim and Ryutov
    - Friday 11 a.m. to noon
  - Chansook Lim
    - Wednesday 10 a.m. to 11 a.m. (regular)
    - Monday 10 a.m. to 11 a.m. (optional - needs notice in advance)

Class Home Page
http://gost.isi.edu/courses/uscd_csci555.html

- Announcements
- Syllabus
- Lecture Slides
- Reading list

Class e-mail: csci555@usc.edu

Administrative Information

- Reading list
  - ~65 papers and
  - ~20 book chapters
  - Concentrated toward the first half

- Text
  - Distributed Systems: Concepts and Design (third edition)
    - By Coulouris, Dollimore, and Kindberg

- Assignments
  - 4 Reports,
    - Due 11 p.m. Thursday nights
  - Research Paper
    - Due: last class
  - Exams
    - Mid-Term: Friday, October 15
    - Final Exam: Friday, December 10

- DEN site - Blackboard
  - Lecture webcast
  - Class forum
  - Grades

- Lecture notes to be posted by Thursdays before lecture
- Academic Integrity
  - READ IT – It applies to you
Administration

- Class forum
  - Announcements
  - Questions
  - Answers
  - Registration
  - Participation

http://den.usc.edu

Administration

- Grading
  - 20%: Reading Reports
  - 20%: Midterm
  - 20%: Final
  - 30%: Research Paper
  - 10%: Class Participation (quiz)

How to survive?

- Read the survival guide
- How to read papers
  - Read the papers in advance
    - Be critical
  - At least skim through
- Build your own notes
- Study group

CSci555: Advanced Operating Systems
Lecture 1 – August 27, 2004

Dr. Dongho Kim
Dr. Tatyana Ryutov
University of Southern California
Information Sciences Institute

What you should learn in this course

- You will gain a basic understanding of distributed system concepts.
- You will develop intuition for which approaches work, and which don’t.
- You will develop the ability to sense where bottlenecks lie in system design.
- You will remember where to look for more information when you are faced with a distributed system problem.
- Above all, you will learn how to be critical of what you are told by system designers.

Some things an operating system does

- Memory Management
- Scheduling / Resource management
- Communication
- Protection and Security
- File Management - I/O
- Naming
- Synchronization
- User Interface
Progression of Operating Systems

Primary goal of a distributed system:
- Sharing

Progression over past years
- Dedicated machines
- Batch Processing
- Time Sharing
- Workstations and PC's
- Distributed Systems

Structure of Distributed Systems

- **Kernel**
  - Basic functionality and protection
- **Application Level**
  - Does the real work
- **Servers**
  - Service and support functions needed by applications
- Many functions that used to be in Kernel now in servers.

Structure of Distributed Systems

- User Space
- Kernel

Network vs. OS Layering

(No direct mapping, colors to stimulate discussion)

- Application Layer
- Presentation Layer
- Session Layer
- Transport Layer
- Network Layer
- Link Layer
- Physical

Characteristics of a Distributed System

- Basic characteristics:
  - Multiple Computers
  - Interconnections
  - Shared State

Why Distributed Systems are Hard

- **Scale**:
  - Numeric
  - Geographic
  - Administrative
- Loss of control over parts of the system
- Unreliability of Messages
- Parts of the system down or inaccessible

- Lamport: You know you have a distributed system when the crash of a computer you have never heard of stops you from getting any work done.
End-to-End Argument

◊ QUESTION: Where to place distributed systems functions?
◊ Layered system design:
  □ Different levels of abstraction for simplicity.
  □ Lower layer provides service to upper layer.
  □ Very well defined interfaces.

E2E Argument (continued)

◊ E2E paper argues that functions should be moved closer to the application that uses them.

E2E Argument (continued)

◊ Rationale:
  □ Some functions can only be completely and correctly implemented with application’s knowledge.
    ◹ Example:
      ─ Reliable message delivery, security
      ─ Encrypted e-mail
      ─ Streaming media vs. Banking
  □ Applications that do not need certain functions should not have to pay for them.

E2E Counter-Argument

◊ Performance
  □ Example: File transfer
    ◹ Reliability checks at lower layers detect problems earlier.
    ◹ Abort transfer and re-try without having to wait till whole file is transmitted.
◊ Abstraction
  □ Less repetition across apps
Bottom line: “spread” functionality across layers.